UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/676,373 | 09/30/2003 | Stefan Jesse | 09700.0216-00 | 3224 |

60668 7590 01/04/2011

SAP / FINNEGAN, HENDERSON LLP
901 NEW YORK AVENUE, NW
WASHINGTON, DC 20001-4413

| EXAMINER | |
|---|---|
| VU, TUAN A | |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2193 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 01/04/2011 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

| Office Action Summary | Application No. | Applicant(s) | |
|---|---|---|---|
| | 10/676,373 | JESSE ET AL. | |
| | Examiner | Art Unit | |
| | TUAN A. VU | 2193 | |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

1)☒ Responsive to communication(s) filed on *11/17/10*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

4)☒ Claim(s) *1, 3-4, 6-10, 12-18, 20--26* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1, 3-4, 6-10, 12-18, 20--26* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

**Application Papers**

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

## DETAILED ACTION

1. This action is responsive to the Applicant's response filed 11/17/10.

As indicated in Applicant's response, claims 1, 10, 12, 18 have been amended. Claims 1, 3-4, 6-10, 12-18, 20--26 are pending in the office action.

### Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1, 3-4, 6-10, 12, 14-18, 23-26 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kadel et al, USPubN: 2002/0184401 (hereinafter Kadel), in view of Severin, USPubN: 2005/0005251 (hereinafter Severin) further in view of Worden, USPubN: 2003/0149934 (herein Worden)

**As per claim 1**, Kadel discloses a computer-readable storage device storing a computer program product for deriving a metadata API from a metamodel in order to develop an application, the computer program product being operable to cause data processing apparatus to:

receive the metamodel in a first language, the metamodel describing a diagram of classes that define the development objects, the development objects representing building blocks for developing the application (e.g. Fig. 3B; para 0117-0126, pg. 8-9 – Note: XIS framework – see Fig. 3A - implementing InfoBean model of source/consumer components and/or meta-relationships expressed within the XIS mediator paradigm, the relationships describing the bean/domain related diagram representation thereof – see: chart diagram, graphical mode - para

0094, pg. 6; Domains, policies, relationships - para 0096-0097, pg. 7-- represented in blocks or sequence diagrams, or domain relationships in relation to consumer/producer interaction – see Fig. 13; UML - see Fig. 8, 11 - **reads on** first metamodel in first language, the first language, for example, as unified modeling language as linked building blocks for developing application -- Fig. 8, 11, 13, 36C-D);

generate a set of intermediate objects to represent the classes of the metamodel by parsing the model description (e.g. Fig. 5; para 0194-0198, pg. 15; JavaBeans ... *package com.xis.leif.event ... includes interfaces* - para 0210-0213, pg. 16; *domain policy methods ... returned ... procedural components of the metadata and methods* - para 0334, pg. 29); and

generate code that is included in the API wherein the API enables development tools to access the development objects (e.g. Add resource class, loads other pluggable services into pluggable manager – Fig. 29; para 0344, pg. 29; *Java interfaces ... instantiate the appropriate* bean -para 0349-0350 pg. 30; exposing attributes, Java introspection - para 0129-0135 pg. 9; JAF/XIS command API – see para 0349-0359, pg. 30; canPaste() Fig. 33b; cut(clipboard) Fig. 33c – Note: using Java introspection class and standard Java API in JAF to generate method or GUI commands for accessing other properties **reads on** creating code in the API enabling development tools to access development objects; see cut and paste - para 0335-0344, pg. 29) to develop the application.

Kadel does not explicitly disclose instructions to convert the metamodel to a model description in a second language according to an interchange format, nor does Kadel explicitly disclose generating intermediate objects by parsing the model description.

Kadel discloses deriving Java objects association with XML constructs or schema (para 0297-0306,pg. 23-25), implementing a "Domain Policy" using XIS framework and DSI interface to expose java objects based on metadata relationship information derived for a domain (see Fig. 13; para 0193, 0203, pg. 4; Fig. 14, 36A-B), relationship to implement the required data flow between source and consumer. Accordingly, Kadel discloses database for assisting the DSI in form of extensible markup schema (e.g. para 0288-0304, pg. 24; para 0084-0085, pg. 5) or descriptive language describing said Domain Policy attributes or typemetadata, attribute relationship or dependency, or declaration constraints; that is, the meta data representing a application domain such that Java related beans – or intermediate Java objects - exposed by the DSI (see para 0311-0312, pg. 26) in Kadel's XIS framework are represented in this XML schema form, which can, in reverse, be **imported back** into a framework (see Kadel: para 0083, pg. 5; *XML schema ... sends to the receiver ... receiver reconstructs the information* - para 0305, pg. 25) its content exposed by the XML-DSI (para 0318, pg. 26) in relation to the database (Fig. 30). Hence the deriving of Java classes based on domain XML as a bi-directional interchange with the corresponding UML is suggested (see Kadel: Fig. 13). The interchangeability of XML and UML in terms of mapping of UML constructs into XML schema and vice versa was well-known and disclosed in Severin. Following to this concept of Kadel's UML/XML inter-changeability, Severin discloses UML constructs (Severin: Fig. 4-8), with use of XML metasyntax and XMI methodology (Severin: XMI - para 0184, pg. 15) to represent this extensible meta language in terms of definitions, inter-relationships or constraints (e.g. Severin: zero-to-many, metahints, relationship, constraint, datatype – see para 0139-0148, pg. 11-12) reading a persisted model (para 0508, pg. 41 ) and re-mapping metamodel data and corresponding UML

package constructs (MVC para 0107-0108, pg. 8; Fig. 32) to derive underlying Java classes or package (para 0196, pg. 16; para 0107-0108, pg. 8). That is, the well-known W3C XMI methodology in terms of data interchange description -- or a second model -- based on a first model (e.g. UML as in Kadel) including model description language to correlate with XML components is evidenced in Severin 's (XML, XMI - para 0184, pg. 15) where rediscovery based on such XMI model description enable remapping into XML elements which are derived for further development; e.g. mapping for developing Java objects or classes, APIs for some domain application. Based on the UML constructs and derived class objects as taught in Kadel's use of the DSI approach and XML processor (XML stream 3002, XML Processor – Fig. 30) the tight association between meta-information and the deriving of Java objects from XML model as shown in Kadel and the XMI implementation as taught in Severin to enable rediscover content of a XML model, it would have been obvious for one skill in the art at the time the invention was made to implement Kadel's XML schema as first metamodel so that a transformation to this model yield a XML-compliant model supported via a XMI (interchange format) whereby exposing the UML instance (see Severin) and underlying Java objects as taught above, because this second model would be used to better collect and identify objects (deriving of UML and underlying Java objects therefrom – see Kadel: Fig. 13; i.e. deriving intermediate objects from that interchange format) exposed from the first model received in a portable schema stream format using the W3C methodology and its useful techniques supporting this XML/model interchangeability as this is also perceived in Kadel, and Severin.

Kadel does not explicitly disclose using the set of intermediate objects as inputs to create API code for enabling access and development objects.

The parsing of model using Java classes or dedicated interfaces or Java APIs was a well-known concept, as evidenced in the XML parsing (Kadel: para 0301, pg. 24) and XIS framework (para 0095, pg. 7) by Kadel, wherein Kadel discloses database for assisting the DSI in form of extensible markup schema (e.g. para 0288-0304, pg. 24; para 0084-0085, pg. 5) or descriptive language describing said Domain Policy attributes or typemetadata, attribute relationship or dependency, or declaration constraints; that is, the meta data representing a application domain such that Java related beans – or intermediate Java objects - exposed by the DSI (see para 0311-0312, pg. 26) in Kadel's XIS framework are represented in this XML schema form, and wherein Kadel uses Java objects association with XML constructs or schema (para 0297-0306,pg. 23-25), to implement a "Domain Policy" using XIS framework and DSI interface as to expose java objects based on metadata relationship information derived for a domain (see Fig. 13; para 0193, 0203, pg. 4; Fig. 14, 36A-B). Using the very Java objects to manage/expose internals of a model as it it is being parsed is disclosed in W3C DOM/SAX methodology in forms of dedicated Java APIs for marshalling, manipulating (or capturing semantic of) schema objects/elements included in on the parsed model, and this is disclosed in **Worden**. Worden discloses the well-known practice of DOM structure representing a schema with associated APIs (Worden: para 0003-0005, pg. 1) where class-model based APIs calls into the XML to access objects therein (Worden: para 0034, pg. 3; *API which ... accesses or creates* - para 0045, pg. 4) to get information (Worden: para 0034, pg. 3) similar to introspection taught in Kadel (see Kadel: para 0129-0135 pg. 9); or to reflect XML meaning (Worden: para 0064, pg. 5; para 0175-0177 pg. 10) in which class or Java APIs are tightly associated with the DOM and in return are available for the developers to create code to manipulate objects or semantics inside the XML schema. It

would have been obvious for one of ordinary skill in the art to implement the capturing of XML schema and parsing of model in Kadel XIS development framework, so that intermediate objects (W3C type Java interfaces) gathered from the process of parsing would be inputs for developing introspection, semantic capturing or schema manipulating code (XML marshalling or re-generating APIs) all of which made available in Kadel's API; because the extensive use of available Java libraries well-known in W3C methodology identified from reading a model within Kadel XIS instance would be used without having to generate source code from scratch, thus facilitating the developers with a pool of APIs to handle the model or the parsed model in terms of analyzing or persisting representation of the model or schema, discovering its content, or using schema semantics (e.g. using intermediate objects, or API's pluggable as input to introspect the parsed objects) so as to implement a target application (as shown in Worden) or to store the model back into its XML interchangeable format as set forth above using Severin's approach.

**As per claim 3**, Kadel discloses wherein the second language comprises XML (refer to the rationale in claim 1 addressing XMI/XML deriving of Java objects).

**As per claim 4**, Kadel discloses wherein the first language comprises UML (refer to claim 1).

**As per claims 6-7**, Kadel discloses wherein the first language comprises a customizable extension (e.g. Fig. 3A; addOneOfNService – Fig. 3B; Fig. 5; 36C-36D; para 0136 pg 9; Fig. 29); wherein the customizable extension is used to implement an additional feature of the API (refer to claim 1 based on addPluginService – Fig. 29).

**As per claim 8**, Kadel does not explicitly disclose wherein the additional feature comprises an indication of a file border. Kadel discloses API for JPanel package that operates on GUI component in terms of resizing, repainting, reshaping, paint Border, set Bounds, set Opaque (see JComponent, awtContainer, awt.Component - Fig. 33E) hence the identification of Gui file border in order to manipulate its graphic content is disclosed. And it would have been obvious for one skill in the art at the time the invention was made to implement the java libraries in view of the user manipulation, so that a feature included in the API would include a file border as set forth from the above, because this would help identify the target file upon which awt operation or painting methods would be defined.

**As per claim 9**, Kadel does not explicitly disclose wherein the API comprises a copy and paste operation. Kadel discloses XIS framework enabling editing of commands on GUI componetns, whereby the user can instantiate operation provided by the JAF API (see para 0349-0359, pg. 30; canPaste() Fig. 33b; cut(clipboard) Fig. 33c). Based on the copy-and-paste nature of the user operations to manipulate metadata attributes pertinent to a source/consumer scenario (see cut and paste - para 0335-0344, pg. 29) and to translate the user-customized parameters in a Java code procedure, it would have been obvious for one skill in the art at the time the invention was made to implement the XIS framework so that metadata and exposed Java classes libraries in view of JAF API (Fig. 33) are combined to support the creation of API type of operation to actually edit the attributes or manipulate exposed meta hierarchy using the standard GUI fabric (e.g. via copy paste functions of GUI components), because this would constitute efficient use of metadata and reusable Java packages whose utilization would be consistent with the extensibility

aspect of the XIS framework, the extensive editing role played by user (Fig. 37A-D), and the availability of JAF API as set forth above.

**As per claim 10**, Kadel discloses a computer-readable storage device storing a computer program product for deriving a metadata API from a metamodel in order to develop an application, the computer program product being operable to cause data processing apparatus to:

receive the metamodel in a first language, the metamodel describing a diagram of classes that define the development objects, the development objects representing building blocks for developing the application(refer to claim 1), wherein the first language comprises unified modeling language(refer to claim 1);

generate a set of intermediate objects to represent the classes of the metamodel (refer to claim 1) by parsing the model description; and

generate code that is included in the API that enables implementation of the development objects, and further wherein the API enables development tools to access the development objects to develop the application (refer to claim 1) wherein the API includes a XML schema that enables implementation of development objects (XML streams - para 0078, pg. 5; para 0047, pg. 2; Fig. 30).

Kadel does not explicitly disclose convert the metamodel to a model description that describes the metamodel in a second language according to an interchange format, wherein the second language comprises XML. But the tight relationship between XML and derived objects represented by UML via a interchange format enabling a bi-directional interchange (using XMI) mechanism whereby UML objects and derived XML objects (or model description) can be

converted from one another, based on the parsing of XML/UML data via the interchange format, has been addressed as obvious in claim 1.

Nor does Kadel explicitly disclose using the set of intermediate objects as inputs to create API code for enabling access and development objects. However, the use of intermediate APIs from parsing a model in light of W3C and XML schema reuse for modeling business process (or development of code within Kadel's API) has been addressed in claim 1.

**As per claim 12**, refer to the rationale in claim 1 and 10 for the XMI/XML limitation.

**As per claim 14**, Kadel discloses wherein the set of intermediate objects comprises Java objects (refer to claim 10).

**As per claim 15**, Kadel discloses (by virtue of XMI, domain schema and XML derivation from XMI, as set forth in claim 10), wherein the XML schema includes a tree based on aggregation relationships in the metamodel (Note: schema derived from original UML reads on tree based on aggregation in the metamodel, itself formulated as UML building blocks modeling language)

**As per claims 16-17**, Kadel does not explicitly disclose wherein the XML schema includes a reference based on an association relationship in the first model, and wherein the XML schema includes a complex type extension based on an inheritance relationship in the first model. UML as shown in Kadel includes association relationship and inheritance relationship (Fig. 3B; para 0080, pg. 5 para 0198-0200, pg. 15) when exposing meta-attributes related to the source/consumer model and data dependency flow. Based on Severin meta integration requiring mapping of complex association with need for new type creation (complex , new type - para 0086, 6; para 0186) among UML type hierarchies, it would have been obvious for one skill in the

art at the time the invention was made to implement the XML schema intended to be reused in a XIS framework so that reference to association relationship to a UML and complex type extension are also represented in order to address the type extension and association needed within defined UML hierarchy, as by the mapping and integration (as in Severin) wherein integreting the schema would need to address complex processes requiring extension into new complex types.

**As per claim 18**, Kadel discloses a computer-readable storage device storing a computer program product for deriving metadata API from a metamodel in order to develop an application, the computer program product being operable to cause data processing apparatus to:

receive the metamodel describing a diagram of classes (refer to claim 1) that define the development objects, the development objects representing building blocks for developing the application(refer to claim 1);

generate a set of intermediate objects to represent the classes of the metamodel (refer to claim 1); and

derive code that is included in the API, wherein the API enables development tools to perform operations on the development objects to develop the application (based on the set of intermediate objects).

Kadel does not explicitly disclose generate an XMI model that is a representation of the metamodel according to an interchange format; nor does Kadel explicitly disclose generating intermediate objects by parsing the XMI model using an XML parser. But the limitation such as a XMI model as interchange format for deriving objects therefrom using a XML parser have been addressed as a combined rationale in claim 1.

Nor does Kadel explicitly disclose derive code included in the API based on the set of intermediate objects, the derived API code to perform oparations on the development objects.

But this has been addressed in claim 1, using Kadel and Worden.

**As per claim 23,** Kadel (by virtue of Severin) discloses wherein the metamodel is stored one storage module (schematized structures -- representing a UML model, imported into a XIS framework – see Kadel: para 0083 pg. 5- reads on UML first model document being stored in a file system of a framework or integration memory)

**As per claim 24**, refer to claim 23.

**As per claims 25-26**, Kadel discloses wherein the set of intermediate objects comprises Java objects (refer to claim 14).

4. Claims 20-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kadel et al, USPubN: 2002/0184401, and Severin, USPubN: 2005/0005251, and Worden, USPubN: 2003/0149934; further in view of Hejlsberg et al, USPN: 6,920,461 (hereinafter Hejlsberg)

**As per claims 20-21**, Kadel discloses operations to include creating a new development object as a transient object without an existing corresponding file (.g. para 0312 pg. 26 – Note: creating of template then use it for adding/deleting Fig. 29-31; Fig. 33 – reads on transient object without a persistent file); but does not explicitly disclose modifying the transient object until the transient object is committed to a persistent file; and to destroy the transient object if a delete command is requested before the transient object is committed to a persistent file.

Kadel discloses code implemented to match SQL queries using XML elements to instantiate application to interface with database (para 0300-0310, pg. 25) where code to implement requires pluggable service and attribute conversion using JAF collaboration of classes

with editing capabilities (Fig. 29-31; Fig. 33) wherein user's deleting, adding of data is effectuated via a template usage (para 0312 pg. 26) all of which data being temporary until determination to commit such implementation, the use of Java interfaces that expose content of a parsed model which is also taught in Worden's API (refer to claim 1). Hejlsberg discloses a development application interface (analogous to Kadel) operating on layers or namespaces that expose class libraries or enumeration of related data structures or code constructs or tables ( see Hejlsberg: col. 6; Fig. 2). Accordingly, Hejlsberg discloses application code instantiation from the libraries of reuse classes or OO packages (e.g. C++, Jscript, Microsoft ".NET" APIs) including UI objects with procedures to save a view, to customize drawing or drag-drop (col. 7, lines 48-62; col 8 lines 22-50) and a SQL namespace to interface with a database (col. 8 line 50 to col 9 line 11) including procedures to validate proper constructs, for implementing operations as to commit, dispose, rollback, save, accept/reject changes, cancel Edit (RejectChanges, acceptChanges – col 55; commit, delete, rollback – col. 57; CancelEdit col. 64; Delete, AcceptChanges – col. 65; Dispose, Finalize – col. 289; Commit, Dispose, Rollback, Save - col. 326). Based on methods based on reuse libraries to implement API in terms of constraints as in Severin and Kadel and the intended framework enabling users to decide whether how to add/remove or commit template/transient data/objects, it would have been obvious for one skill in the art at the time the invention was made to implement the code or APIs based on core libraries as practiced in both Kadel and Hejlsberg, such that a transient object in the process of validating data, information, and implementation details as taught in Kadel's user-driven customization approach (e.g. using template) would be supported by capability to create APIs with methods to destroy a transient object or to commit it to a persistent form, as taught in

Hejlsberg from above. One would be motivated to do this (i.e. create APIs by the user to destroy a transient object if it is not made for persistence committing) because that way the created APIs would enable changes caused by a customization view in Kadel's approach to detect errors prior to commit, and allowing removal of undesired implementation gathering of data, whereby obviate potential runtime errors should actual translation of uncorrected constructs become finalized.

**As per claim 22**, Kadel does not explicitly disclose instructions to mark the persistent file as deleted if a delete command is requested after the transient object is committed to a persistent file. Based on Hejlsberg's DB-related method to indicate that change data is not accepted or that a transient form of changes is committed, or to rollback otherwise (e.g. RejectChanges, acceptChanges – col 55; commit, delete, rollback – col. 57), the notion of keeping a change with persisting of a accepted version along with removing an older version is suggested. Hence, this method as to mark an older file/record as deleted after a persisting operation has been completed (by creating a new file) would have been obvious in view of the requirement to reconcile persisted data (e.g. DB records, not keeping two records with same identification) rationale as set forth above.

**Response to Arguments**

5. Applicant's arguments filed 11/17/10 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

(A) Applicants have submitted that Kadel source component and data consumer component cannot constitute a API that enables development tools to access … objects (Applicant's Remarks pg. 9, top). "Development tools" have been broadly interpreted and thus includes

software plug-ins, APIs or W3C type of libraries classes or DOM interfacing codes. The API has been intepreted as a development interface that provide all such development tools. The argument is deemed not sufficient to overcome the teachings proferred based on Kadel, mainly because the claim language does not include compelling details as to preclude application of that reference.

(B) Applicants have submitted that Kadel and Severin fail to teach generating API with code included in the API, "using the set of intermediate objects as inputs" (Applicant's Remarks pg. 9 middle). The newly amended language has necessitated a new ground of rejection, the argument becoming largely moot.

(C ) Applicants have submitted that Kadel, Severin, Hejlsberg fail in combination to teach claims 20-22 because the references cannot remedy to the deficiencies mentioned above (Applicant's Remarks pg. 10). The maintained rejection state of these claims rests on the non-persuasiveness of the arguments as presented in the above sections.

In all, the claims stand rejected as set forth above.

**Conclusion**

6. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 ( for non-official correspondence - please consult Examiner before

using) or 571-273-8300 ( for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

January 03, 2011